Processamento de Linguagens I
LESI + LMCC (3º ano)

Exame de 2ª Época

Data: 07 de Setembro de 2002
Hora: 09:30

Dispõe de 2:00 horas para realizar este exame

Questão 1: Escrita de uma Gramática

Um arquitecto pretende formalizar uma linguagem para escrever *memórias descritivas dos seus projectos*.
Em primeiro lugar, a linguagem deve permitir identificar a obra—quem é o dono, onde se localiza e qual o valor total do orçamento—e a data de conclusão do projecto. Depois, devem indicar-se os andares; para cada andar, serão indicadas todas as divisões, dizendo-se para cada uma o seu tipo (a sua serventia) e a área, bem como a eventual existência de um armário embutido. A seguir descrevem-se os anexos, casos existam, fornecendo para cada um informação semelhantes às demais divisões. Por fim, indica-se a área toda de jardim, especificando se é a toda a volta, ou só em alguns dos lados.

Responda, então, às seguintes alíneas:

a) Construa uma gramática independente de contexto (GIC) que defina a linguagem pretendida, de modo a ser usada para construção de um *processador bottom-up* (por exemplo, recorrendo ao gerador yacc) eficiente.

b) Mostre, justificando, que alterações faria à gramática caso tivesse de implementar um *processador top-down*.

Questão 2: Gramáticas, Linguagens e Parsing Top-Down

Analise atentamente a seguinte GIC, cujo Símbolo Inicial é `Prog` e em que todos os símbolos escritos em minúsculas são pseudo-terminais, as palavras-reservadas são escritas em maiúsculas e os sinais estão entre apostrofes.

```
Prog      -->  Cabec Corpo Fim
Cabec     -->  PROG id ';'
Corpo     -->  Decls Insts
Decls     -->  &
Decls     -->  Decls Dcl
Dcl       -->  CONST id '=' val ';'
Dcl       -->  VAR   id ':' id ';'
Insts     -->  Cmd
Insts     -->  Insts ';' Cmd
Cmd       -->  c1
Cmd       -->  c2
Cmd       -->  c3
Fim       -->  '.'
```

Responda, então, às alíneas seguintes:

a) Explique por palavras suas qual é a linguagem gerada por essa GIC.

b) Dê um exemplo de uma frase válida dessa linguagem, apresentando a respectiva árvore de derivação.

c) Olhando directamente para a forma das produções, é imediato concluir que *a gramática apresentada não é LL(1), visto ser recursiva à esquerda.* Prove que essas situações de recursividade à esquerda implicam a não verificação da condição LL(1).

d) Recorrendo à linguagem de programação C, escreva as rotinas dum *Parser Recursivo Descendente* correspondentes ao reconhecimento dos símbolos não-terminais `Prog` e `Cmd` e do símbolos terminais (um só procedimento para todos).

Questão 3: Gramáticas e Parsing Bottom-Up

Considere a seguinte versão reduzida da gramática anterior:

```
Prog      -->  Cabec Decls Fim
Cabec     -->  PROG id ';'
Decls     -->  &
Decls     -->  Decls Dcl ';'
Dcl       -->  CONST id '=' val
Dcl       -->  VAR   id ':' id
Fim       -->  '.'
```

a) Verifique se a gramática é LR(0), construindo o autómato LR(0) e as respectivas tabelas de parsing *bottom-up.*

b) Especifique um analisador sintáctico para as frases da linguagem gerada por essa gramática, usando o yacc.

c) Acrescente à especificação yacc anterior acções semânticas para contar o número de variáveis declaradas do tipo "int"e para escrever, no fim do parsing, a lista das constantes declaradas por ordem alfabética (suponha que as funções auxiliares de que precisa já estão implementadas).

Questão 4: filtros de texto

Usando o gerador de programas flex descreva um **filtro de texto** para resolver cada um dos seguintes problemas:

a) Pretende-se um filtro que reproduza na saída um texto de entrada retirando-lhe tudo o que estiver intercalado entre as marcas '`++cod++`' e '`--cod--`', ou então, pelo contrário, retire todo o restante texto, deixando apenas o que estiver entre essas duas marcas. A escolha, ou modo de funcionamento depende de se encontrar no início de uma linha o comando '`++limpa++`' (activa o 1º caso) ou o comando '`++extrai++`' (corresponde ao 2º caso) .

b) Suponha que um ficheiro de texto é formado por linhas todas com a mesma estrutura: 4 campos de informação— `codigo da disciplina` (6digitos), `nome da disciplina`, `codigo do departamento responsável` (3 dígitos), e `número de inscritos para exame`— separados pelo caracter '`;`'.
Pretende-se um processador de texto que copie para a saída o nome e número de inscritos apenas às disciplinas da LESI (código começado por `530`) leccionadas pelo DI (código `406`).